An Inverse Procedural Modeling Pipeline for Stylized Brush Stroke Rendering

Hao Li*, Zhongyue Guan*, Zeyu Wang

The Hong Kong University of Science and Technology (Guangzhou), China

Abstract

Stylized brush strokes are crucial for digital artists to create drawings that express a desired artistic style. To obtain the ideal brush, artists need to spend much time manually tuning parameters and creating customized brushes, which hinders the completion, redrawing, or modification of digital drawings. This paper proposes an inverse procedural modeling pipeline for predicting brush parameters and rendering stylized strokes given a single sample drawing. Our pipeline involves patch segmentation as a preprocessing step, parameter prediction based on deep learning, and brush generation using a procedural rendering engine. Our method enhances the overall experience of digital drawing recreation by empowering artists with more intuitive control and consistent brush effects.

CCS Concepts

• ***Computing methodologies*** → ***Image-based rendering;*** *Machine learning;* • ***Applied computing*** → *Media arts;*

1. Introduction

Recent advances in digital technologies have enabled artists to use a set of digital tools for artistic creation. In particular, stylized brush strokes play a pivotal role in creating digital drawings that express certain artistic effects [LBDF13, SLF22]. However, most of these methods focused on example-based or patch-based rendering that can barely support real-time rendering such that the output brushes cannot be easily used by the artists. In practice, stamp-based brushes are dominant in common paint software, which this work particularly focuses on. Professional artists often customize these brushes by tuning parameters in the brush engine to enhance their style and finetune their artwork. The creation and selection of brushes represent a crucial step for digital art creators in the drawing process.

A common challenge faced by artists is to create stylized brushes that match the existing brush styles in other drawings. Despite their ability to mimic suitable brush effects by studying existing artworks, artists frequently encounter difficulties because they often only have a few reference images about the ideal brush and find it difficult to locate corresponding brush resources that can be directly used for drawing. It is also cumbersome to create a desired brush, as this step is time-consuming and requires expertise to identify, summarize, and adjust brush parameters [Her98]. For novice painters, this situation becomes more challenging, inhibiting them from employing digital drawing techniques.

In this paper, we proposed a new pipeline that can automatically predict the parameters of stamp brushes and reconstruct the digital brushes involved in a single stylized line drawing. Given a sketch, we developed a segmentation algorithm to identify the different brush styles as patches. Then we designed a convolutional neural network to predict the parameter sets from those patch images. We trained our network with our large-scale generated dataset of stroke patches and their registered parameters. With the predicted parameter set, we are able to reconstruct the stamp brush using the open-source paint software *Ciallo* [CW23]. This simplifies the process for artists to collect suitable brushes, assisting them in inheriting the original painting style and brushstrokes, making it more convenient and effective.

In summary, our paper makes the following contributions:

- A stroke segmentation method to identify different styles of strokes and generate corresponding patches for parameter prediction.
- A dataset of more than 50K individual stylized stroke patches with registered brush parameters.
- A convolutional neural network that can predict stamp brush parameters to match the style in a given stroke patch image.

2. Methodology

Our core workflow is shown in Figure 1. We explain the segmentation algorithm in Section 2.1, the parameterized stroke dataset in Section 2.2, and the prediction model in Section 2.3.

*Equal contribution.

DOI: 10.2312/egs.20241024

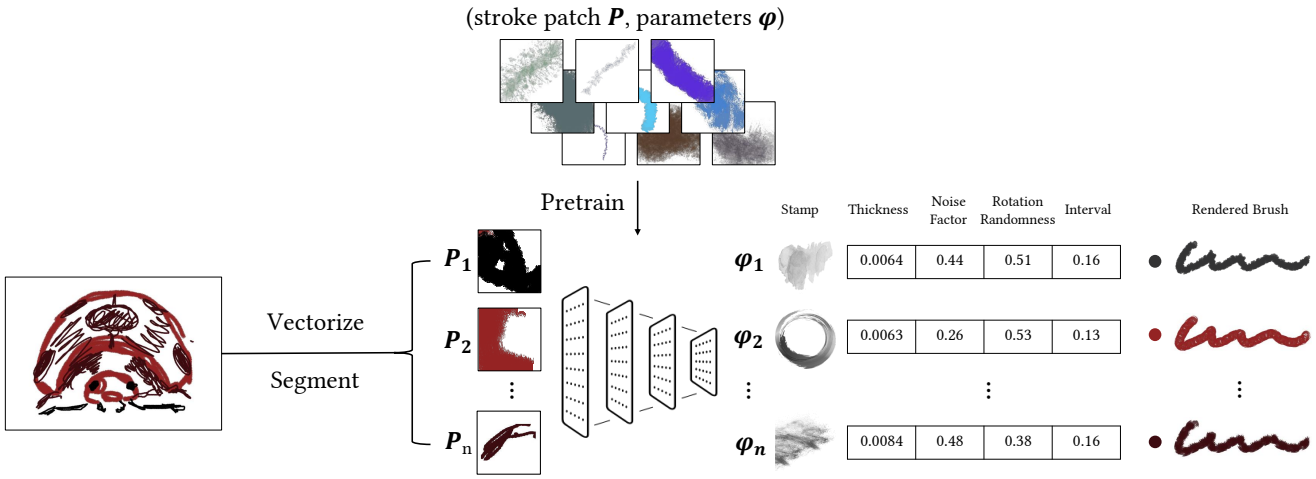

Figure 1: *Our proposed pipeline. Given a reference drawing, we perform stroke segmentation to obtain stroke patches. Then we train a convolutional neural network to predict stroke parameters, with which we can generate similar brush strokes.*

2.1. Stroke-based Patch Segmentation

Current brushstroke segmentation methods, such as DStroke proposed by Fu et al. [FYY*21], tackle with paintings with densely overlapped strokes. Different from their focus, we particularly consider sketchy line drawings where sparse and long strokes are used to explicitly depict shape and texture.

Initially, we used the Canny algorithm [Can86] for recognizing edges in sketches, and then applied dilation and Gaussian blur to smooth the brush edges, but it was less effective for denser sketches. The illustration can be found in supplementary material.

Observing that brush effects are rendered following drawing trajectory in common paint software, we considered the stroke path as critical information for stroke segmentation of line drawings. We prioritized obtaining path information from the user input side, leveraging the data provided by existing sketch dataset [GSH*19, WQF*21, XSL*22]. And we also tried to vectorize sketches using raster-to-vector technique [Sel03].

Given the stroke path information, a bounding box can be extracted for each individual stroke, from which the overlaps among all brush strokes in the drawing can be further calculated. We selected regions with less stroke overlap to favor single stroke with explicit brush features. These identified regions are considered effective for assisting in brush feature extraction and recognition that are subsequently cut from the original digital artwork.

Through the previous process, single segmented patch may still include more than one brush effects. Therefore, we designed a brush segmentation algorithm that analyzes all pixels in the patch, further segmenting the region based on pixel gradient variations. Algorithm 1 summarizes the steps of this approach. Finally, we cropped each segmented patch at the ratio of 1:1 and then normalized each of them to be (224, 224) resolution to fit in the network input.

2.2. Stroke Patch Dataset

We collected the stroke patch dataset by rendering diverse stamp-based brushes on given vector paths. We generated 6,677 sets of parameters φ = (stamp image, interval, thickness, rotation randomness, noise factor), which are the basic set of properties to define a stamp-based brush in common paint software such as Photoshop. For the stamp image, we collected 107 public ones with various

Algorithm 1 Stroke Segmentation

```
image ← Input image matrix
threshold ← Pixel value threshold that triggers segmentation
height, width ← dimensions of image
regionSet ← empty list to store strokes
function DFS(x, y)
    oriColor ← image[y, x]
    Add [y, x] to the last list in regionSet
    Initialize stack with adjacent pixels of (x, y)
    while stack is not empty do
        (curX, curY) ← pop an element from stack
        curColor ← image[curY, curX]
        if abs(curColor - oriColor) < threshold then
            Update oriColor
            Add [curY, curX] to the last list in regionSet
            Update stack with adjacent pixels of (curX, curY)
        end if
    end while
end function
for each pixel (x, y) in image do
    if pixel (x, y) is not visited and not black then
        Add new list to regionSet
        DFS(x, y)
    end if
end for
return regionSet
```

brush effects from an online platform [PNG23]. We selected 8 vector paths with simple line topology and distinct curvature since we also favor single stroke without overlapping in the segmentation algorithm. Each vector path is defined as a polyline $L = (x_i, y_i)_{i=0}^{n}$. Given each set of parameters, we rendered the brushes on given vector paths using the open-source paint software *Ciallo* [CW23], resulting in RGBA images at (224, 224) resolution. We converted them to RGB images to align with the segmented patch from stylized line drawings. Therefore, we generated $6,677 \times 8 = 53,416$ pairs of (stroke patch P, parameters φ) as our training data.

2.3. Brush Prediction

We formulated the prediction of stamp images as a classification problem and the prediction of other parameters as linear regression problems. We tried several convolutional neural network architectures and settled on the best-performing one that follows the ResNet-18 architecture. We changed the fully connected layer to have 111 outputs that contains predicted probabilities for each of the 107 stamp images and predicted values for interval, thickness, noise factor, and rotation randomness. The architecture details can be found in the supplementary material.

Data Pre-processing. We augmented the dataset by randomly rotating the patch image in order to enhance the variety of collected stroke patches. In addition, we applied min-max normalization to the brush parameters to equal the contribution of each parameter to the model.

Training. We used 64:16:20 for the train-validation-test split. Each patch is treated as a separate training instance. We use six terms in our loss function, which is formulated as follows:

$$\mathcal{L} = \lambda_s \mathcal{L}_s + \lambda_i \mathcal{L}_i + \lambda_n \mathcal{L}_n + \lambda_r \mathcal{L}_r + \lambda_t \mathcal{L}_t$$

The hyperparameters are set to the default value 1. The detailed explanation of the loss terms can be found in the supplementary material.

We trained the network from scratch using the Adam optimizer with learning rate set to 10^{-4} and batch size 64 on an NVIDIA GeForce RTX 4090 GPU. It takes about 5 hours on average to converge in our experiments.

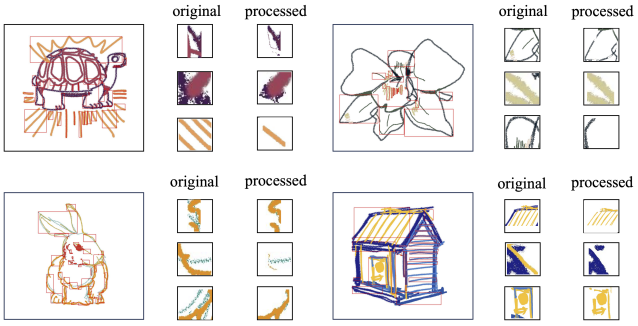

Figure 2: *Patch segmentation results. Given an existing drawing and its vectorization information, our algorithm can extract the bounding boxes of strokes and then process the segmented patches to only keep one primary stroke style in each.*

3. Results

In this section, we demonstrate some experimental results of patch segmentation and brush parameter prediction.

Patch Segmentation. Figure 2 shows the patch segmentation results. Given an existing drawing and its vector path, our algorithm first extracts the bounding boxes of strokes. Then we process it to keep only one primary stroke style in each patch. As shown in the figure, the patterns of strokes that can be obviously observed are well captured in the segmented patches. After further processing, the individual patch is capable of expressing the captured feature of one specific brush.

Brush Prediction. We report our evaluation results on the test set in Table 1. Our model can make decent predictions with an accuracy of 90.17% for stamp image, and quite low mean square errors for interval, thickness, noise factor, and rotation randomness.

Table 1: *Prediction accuracy and errors.*

Stamp Image (%)	Thickness (10^{-4})	Interval (10^{-4})
90.17	0.45	1.86

Noise Factor (10^{-4})	Rotation Randomness (10^{-4})
5.85	12.35

To evaluate the performance of our entire pipeline, we present the brush reconstruction results from stylized drawings in Figure 3. Our method can restore a similar brush effect in terms of overall appearance by segmenting stroke patches from the whole drawing and predicting individual brush parameters with our trained model.

4. Limitations and Future Work

There are several limitations of our proposed approach. We only considered the strokes with fixed width while omitting the influence of pen pressure. One potential direction is to infer thickness offset and the mapping between pen pressure and stroke thickness based on a group of strokes with varying widths. Moreover, the brush reconstruction in this study is currently restricted to reproducing stamp brushes with a bald parameter set. To improve practicality, it is crucial to predict complex brushes with diverse shapes, textures, and even dynamic simulation parameters within a single stroke. We will explore the potential of developing an end-to-end differentiable pipeline by considering the intended brush as a function. Achieving this goal requires a systematic consolidation of artist experiences and traditional brush effects.

5. Conclusion

In this paper, we address the challenge of acquiring user-intended brushes in the creative process and optimizing artists' drawing experience by presenting a novel pipeline for intelligent stroke extraction, parameter prediction, and stamp brush generation. Our work has the potential to be seamlessly integrated into common paint software.

We incorporated a segmentation algorithm and neural network for inverse procedural modeling of stylized brush strokes to enhance and streamline the basic drawing tools commonly used by

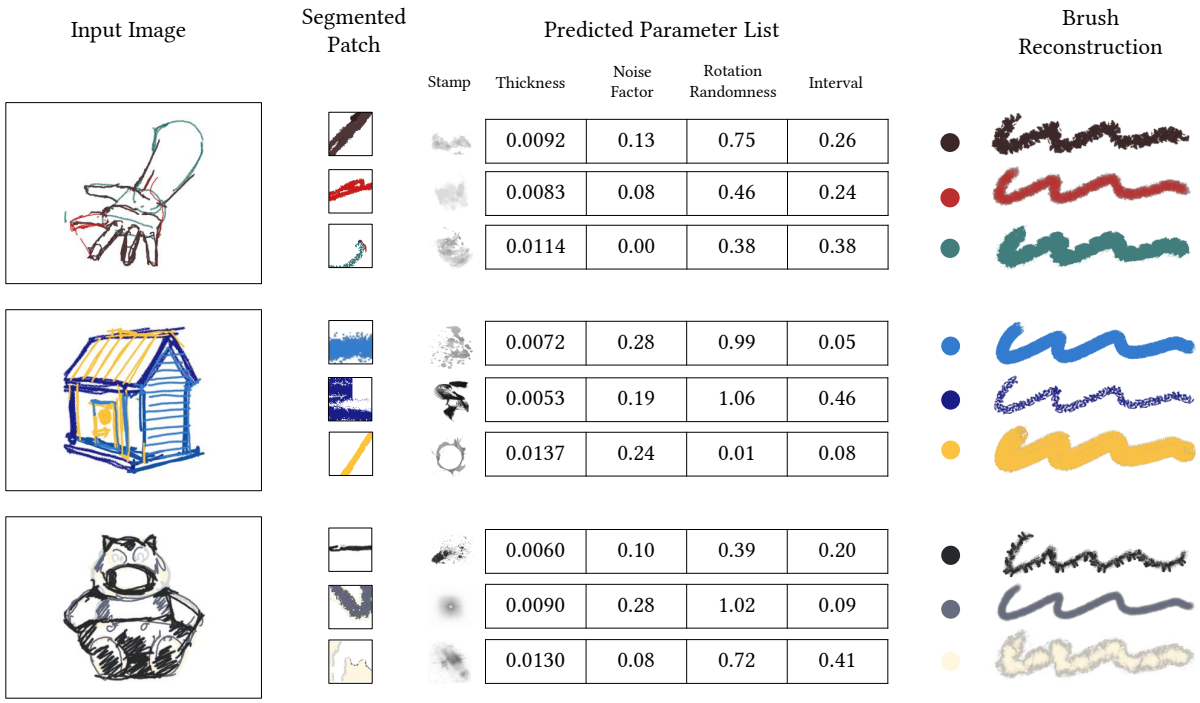

Figure 3: *Brush reconstruction results from stylized drawings. The majority of brushes generated from the predicted parameters can resemble the input patches. In the last row, we found an unexpected result and concluded it as the potential gap between the segmentation algorithm and the prediction model. Despite the stroke color of ivory, our model tried to mimic the gray boundary observed in the patch, which is actually caused by the stroke filtering in the segmentation algorithm.*

artists. By providing a single stylized line drawing as a reference, the user can obtain usable stamp brushes that replicate the effects seen in the reference image using our pipeline. The generated brushes can still be further tuned by adjusting individual parameters as the user wants. We validated this novel workflow for intelligent brush generation with diverse digital line drawings.

We believe this work tackles some challenges in brush-based digital art creation and can inspire future intelligent drawing systems.

References

[Can86] CANNY J.: A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (1986), 679–698. 2

[CW23] CIAO S., WEI L.-Y.: Ciallo: The Next-Generation Vector Paint Program. In *ACM SIGGRAPH 2023 Talks*. 2023, pp. 1–2. 1, 3

[FYY*21] FU Y., YU H., YEH C.-K., LEE T.-Y., ZHANG J. J.: Fast Accurate and Automatic Brushstroke Extraction. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 17*, 2 (2021), 1–24. 2

[GSH*19] GRYADITSKAYA Y., SYPESTEYN M., HOFTIJZER J. W., PONT S., DURAND F., BOUSSEAU A.: OpenSketch: A Richly-Annotated Dataset of Product Design Sketches. *ACM Trans. Graph. 38*, 6 (nov 2019). URL: https://doi.org/10.1145/3355089.3356533, doi:10.1145/3355089.3356533. 2

[Her98] HERTZMANN A.: Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, Association for Computing Machinery, p. 453–460. URL: https://doi.org/10.1145/280814.280951, doi:10.1145/280814.280951. 1

[LBDF13] LU J., BARNES C., DIVERDI S., FINKELSTEIN A.: Realbrush: Painting with Examples of Physical Media. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12. 1

[PNG23] PNGEGG: Free Transparent PNG Images. https://www.pngegg.com/, 2023. Accessed on: December 30, 2023. 3

[Sel03] SELINGER P.: Potrace: A Polygon-Based Tracing Algorithm, 2003. 2

[SLF22] SHUGRINA M., LI C.-Y., FIDLER S.: Neural Brushstroke Engine: Learning a Latent Style Space of Interactive Drawing Tools. *ACM Transactions on Graphics (TOG) 41*, 6 (2022), 1–18. 1

[WQF*21] WANG Z., QIU S., FENG N., RUSHMEIER H., MCMILLAN L., DORSEY J.: Tracing Versus Freehand for Evaluating Computer-generated Drawings. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–12. 2

[XSL*22] XIAO C., SU W., LIAO J., LIAN Z., SONG Y.-Z., FU H.: DifferSketching: How Differently Do People Sketch 3D Objects? *ACM Trans. Graph. 41*, 6 (nov 2022). URL: https://doi.org/10.1145/3550454.3555493. 2